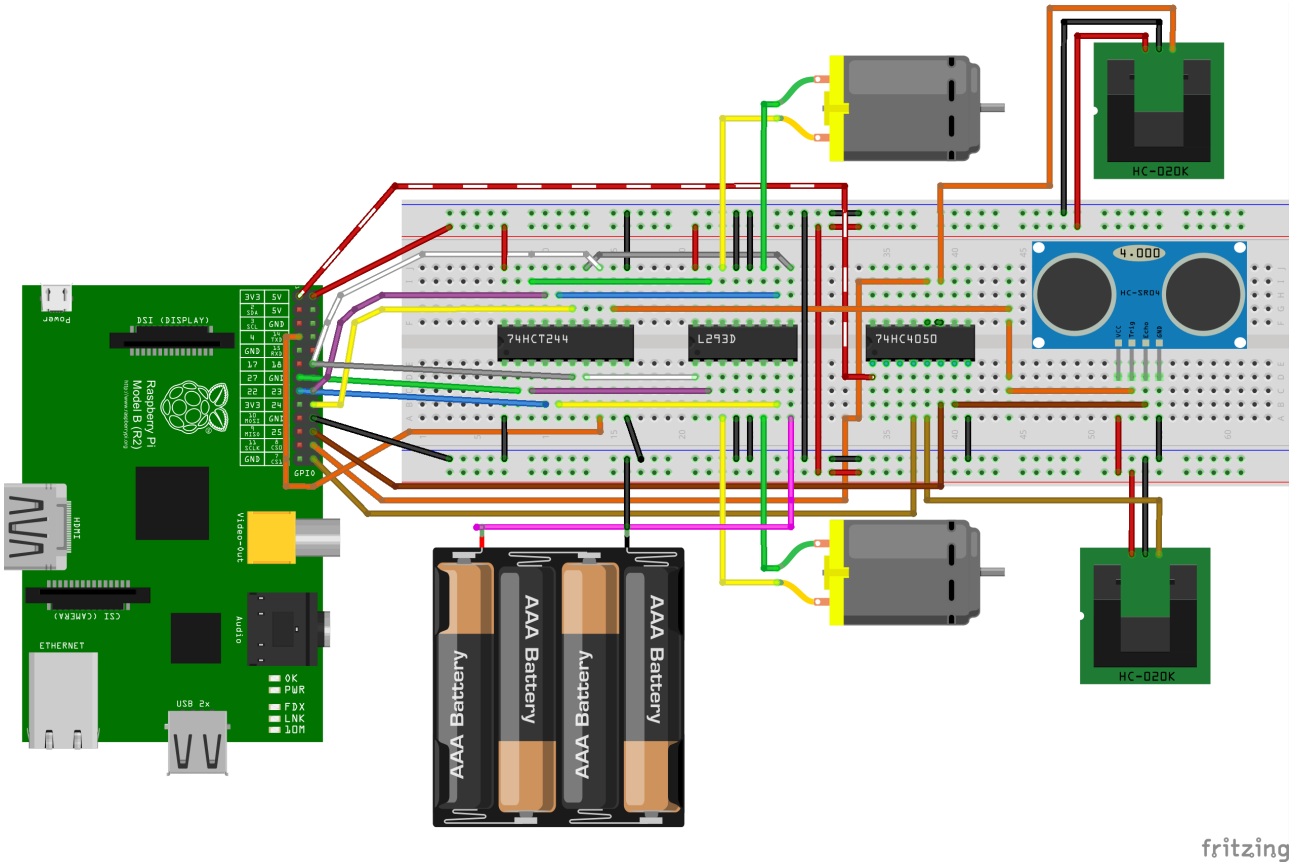
Raspberry Pi – Robotersteuerung

Motorsteuerung und Entfernungsmessung via GPIOs

Benötigte Teile:

1 x Roboterbausatz mit 2 DC-Motoren
2 x HC-020K Drehzahlgeber
1 x 74HCT244 (Pegelwandler 3,3 V zu 5 V)
1 x L293D (Motortreiber)
1 x WLAN – USB-Adapter
1 x 74HC4050 (Pegelwandler 5 V zu 3,3 V)
1 x HC-SR04 Ultraschall Entfernungssensor
4 x AA-Batterien (Motorversorgung)
1 x USB-Powerbank (Raspberry Pi Versorgung)
1 x Raspberry Pi Kamera



encoder.c : Zurückgelegte Distanz ermittelt über Drehzahlgeber der Räder

```
//
//   measure rpm of drive encoder on GPIO 7 and 8
//
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#include <sys/time.h>

static sig_atomic_t end = 0;
const int GPIO_LEFT  = 8;
const int GPIO_RIGHT = 7;
const int PulsPerRotation = 20;
const float circumfer = 0.216; //meter

volatile int nCounterLeft = 0;
volatile int nCounterRight = 0;

//Interrupt Service Routine for GPIOs
void ISR_GPIO_LEFT(void) {
        nCounterLeft++;
}
void ISR_GPIO_RIGHT(void) {
        nCounterRight++;
}

static void sighandler(int signo) {
        end = 1;
}

int main(void) {
        struct sigaction sa;
        double fTime, fFrequencyLeft, fFrequencyRight;
        struct timeval TimeValue, TimeValueOld;
        int nCountLeft, nCountLeftOld;
        int nCountRight, nCountRightOld;

        nCountLeft = 0;
        nCountRight = 0;
        TimeValue.tv_sec = 0;
        TimeValue.tv_usec = 0;
        if (wiringPiSetupGpio() == -1) {
                printf("wiringPiSetup failed\n\n");
                exit (EXIT_FAILURE) ;
        }

        memset(&sa, 0, sizeof(struct sigaction));
        sa.sa_handler = sighandler;
        sigaction(SIGINT,  &sa, NULL);
        sigaction(SIGQUIT, &sa, NULL);
        sigaction(SIGTERM, &sa, NULL);

// INT_EDGE_BOTH, INT_EDGE_FALLING, INT_EDGE_RISING only one ISR per input
        if (wiringPiISR(GPIO_LEFT, INT_EDGE_FALLING, &ISR_GPIO_LEFT)  < 0) {
                printf("Unable to setup ISR for GPIO %d (%s)\n\n",
                  GPIO_LEFT, strerror(errno));
                exit(EXIT_FAILURE);
        }
        if (wiringPiISR(GPIO_RIGHT, INT_EDGE_FALLING, &ISR_GPIO_RIGHT)  < 0) {
                printf("Unable to setup ISR for GPIO %d (%s)\n\n",
```

```
                GPIO_LEFT, strerror(errno));
            exit(EXIT_FAILURE);
        }

        while (!end) {
            nCountLeftOld = nCountLeft;
            nCountRightOld = nCountRight;
            TimeValueOld = TimeValue;
            delay(1000);
            gettimeofday(&TimeValue, 0);
            nCountLeft = nCounterLeft;
            nCountRight = nCounterRight;
            if (0==TimeValueOld.tv_sec && 0==TimeValueOld.tv_usec) {
                continue;
            }
            fTime = (TimeValue.tv_sec - TimeValueOld.tv_sec) +
             (TimeValue.tv_usec - TimeValueOld.tv_usec)/1000000.0;
            fFrequencyLeft = (double)(nCountLeft - nCountLeftOld) / fTime;
            fFrequencyRight = (double)(nCountRight - nCountRightOld) / fTime;
            printf("IntCount: %d/%d, Freq.: %.1f/%.1f Hz, Upm %.0f/%.0f, dis.: %.1f/%.1fm\n",
             nCountLeft - nCountLeftOld, nCountRight - nCountRightOld , fFrequencyLeft,
             fFrequencyRight, fFrequencyLeft*60.0/PulsPerRotation, fFrequencyRight*60.0/PulsPerRotation,
             nCounterLeft*circumfer/PulsPerRotation, nCounterRight*circumfer/PulsPerRotation );
        }

        return(EXIT_SUCCESS);
}
```

HC-SR04.c: Hinderniserkennung mit Ultraschalldistanzmessung (Sensor HC-SR04):

```
//
//  Simple program to measure distance with HC-SR04 sensor
//
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#include <sys/time.h>
#include <unistd.h>

static sig_atomic_t end = 0;
const int GPIO_TRIG = 4;  //TRIG = GPIO4  [out]
const int GPIO_ECHO = 25; //ECHO = GPIO25 [in]
const double SPEED_SOUND = 343.0; // m/s

volatile int nCounterEdgeDown = 0;
volatile int nCounterEdgeUp   = 0;
volatile struct timeval starttime, endtime;

//Interrupt Service Routine for ECHO input
void ISR_ECHO(void) {
	struct timeval now;
	int nInputState;

	gettimeofday(&now, NULL);
	nInputState = digitalRead(GPIO_ECHO);
	if (HIGH ==  nInputState) {
		starttime = now;
		nCounterEdgeUp++;
	} else {
		endtime = now;
		nCounterEdgeDown++;
	}
}

static void sighandler(int signo){
	end = 1;
}


int main(void) {
	struct sigaction sa;
	double secs_elapsed, distance;

	if (wiringPiSetupGpio() == -1) {
		printf("wiringPiSetup failed\n\n");
		exit (1) ;
	}

	memset(&sa, 0, sizeof(struct sigaction));
	sa.sa_handler = sighandler;
	sigaction(SIGINT,  &sa, NULL);
```

```
	sigaction(SIGQUIT, &sa, NULL);
	sigaction(SIGTERM, &sa, NULL);

	pinMode(GPIO_TRIG, OUTPUT);

//INT_EDGE_BOTH, INT_EDGE_FALLING, INT_EDGE_RISING only one ISR per input
	if (wiringPiISR(GPIO_ECHO, INT_EDGE_BOTH, &ISR_ECHO)  < 0) {
		printf("Unable to setup ISR for GPIO %d (%s)\n\n",
		  GPIO_TRIG, strerror(errno));
		exit(1);
	}

	while(!end) {
		//printf("start impuls ...\n");
		digitalWrite(GPIO_TRIG, HIGH);
		usleep(20000);
		digitalWrite(GPIO_TRIG, LOW);
		usleep(80000);

		secs_elapsed = (endtime.tv_sec - starttime.tv_sec) +
		  (endtime.tv_usec - starttime.tv_usec)/1000000.0;
		distance = SPEED_SOUND * 100 * secs_elapsed / 2; //cm
		printf("distance: %.1f cm (time elapsed: %g s)\n",
		  distance, secs_elapsed);
		usleep(250000);
	}

	digitalWrite(GPIO_TRIG, LOW);
	pinMode(GPIO_TRIG, INPUT);
	return 0;
}
```